

## 5\_3\_Magic\_BLAST\_contigs\_specific

August 6, 2021

Summary of magicblast

```
[1]: import numpy as np
import os.path
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
from pandas.plotting import scatter_matrix
import pathlib
import warnings
warnings.filterwarnings('ignore')
```

```
[2]: from IPython.core.display import display, HTML
display(HTML("<style>.container { width:95% !important; }</style>"))
```

<IPython.core.display.HTML object>

```
[3]: PROJECT_CODE='PRJNA606875'
BASE_PATH = f'/mnt/1TB_0/Data/Assembly/{PROJECT_CODE}/'

dbname='nt'
kmer='k141'
kmer_extra='k29'

#magicblast on contigs
magic_blast_sam_tail=f'_{dbname}_final_contigs_magicBLAST.sam'
f_contigs_file_tail=f'_{dbname}_magic_blast_asc_contigs.txt'
GIS,ACCESSIONS,TITLES=None,None,None
```

### 0.0.1 All SRA's

```
[4]: sra_list=['SRR11093265',
↳ 'SRR11093266', 'SRR11093267', 'SRR11093268', 'SRR11093269', 'SRR11093270', 'SRR11093271']

[5]: def read_gi_accession_title(gi_asc_file):
    gis=[]
    accessions=[]
    titles=[]
```

```

with open(gi_asc_file, 'r') as infile:
    data = infile.readlines()
    for i in data:
        output=i.split(' ',2)
        gis.append(output[0])
        accessions.append(output[1])
        titles.append(output[2])
return gis, accessions, titles

```

```

[6]: def set_accessions():
    global GIS
    global ACCESSIONS
    global TITLES
    GIS,ACCESSIONS,TITLES=read_gi_accession_title('/mnt/1TB_ssd/Data/BLAST/nt.
↳gi_taxid.tsv')

```

## 0.0.2 Stats

```

[7]: def get_asc_descr_count(sra):
    accessions=[]
    descriptions=[]
    counts=[]
    path = BASE_PATH+sra+'/magic_blast/'
    f_mer=path+f'{sra}_{dbname}_{kmer}_magicBLAST_summary.txt'
    if not os.path.isfile(f_mer):
        f_mer=path+f'{sra}_{dbname}_{kmer_extra}_magicBLAST_summary.txt'
    assert os.path.isfile(f_mer)
    with open(f_mer, 'r') as f:
        lines = [line.rstrip('\n') for line in f]
        total=len(lines)-1
        for line in lines:
            if not 'database:' in line: #ignore header
                asc=line.split(None, 1)[0]
                title=line.split(None, 1)[1].split(', count:')[0]
                count=line.split(None, 1)[1].split(', count:')[1]
                accessions.append(asc)
                descriptions.append(title)
                counts.append(int(count))
    return accessions, descriptions, counts, total

```

```

[8]: def get_indexes(substr, str_list):
    index_list = []
    i = 0
    for e in str_list:
        if substr in e.lower():
            index_list.append(i)
        i +=1

```

```
return index_list
```

```
[9]: for i,sra in enumerate(sra_list):  
      accessions, descriptions, counts, total=get_asc_descr_count(sra)  
      print(f'{sra}, matched contigs: {sum(counts)}')
```

```
SRR11093265, matched contigs: 44  
SRR11093266, matched contigs: 549  
SRR11093267, matched contigs: 2056  
SRR11093268, matched contigs: 189  
SRR11093269, matched contigs: 1698  
SRR11093270, matched contigs: 19067  
SRR11093271, matched contigs: 156
```

```
[10]: def get_desc_count(qstring, descriptions, counts, lowercase=True):  
       qd=[]  
       qc=0  
       for d,c in zip(descriptions, counts):  
           if lowercase:  
               if qstring.lower() in d.lower():  
                   qd.append(d)  
                   qc=qc+int(c)  
           else:  
               if qstring in d:  
                   qd.append(d)  
                   qc=qc+int(c)  
       return qd, qc
```

```
[11]: def get_descr(sra):  
       accessions, descriptions, counts, total=get_asc_descr_count(sra)  
       print(f'sra: {sra}, accessions: {len(accessions)}, descriptions:␣  
→{len(descriptions)}, counts: {len(counts)}, total: {total}')  
       if total>0:  
           human,humanc = get_desc_count('human', descriptions, counts)  
           homo_sapiens,homo_sapiensc = get_desc_count('homo sapiens',␣  
→descriptions, counts)  
           h_sapiens,h_sapiensc = get_desc_count('h.sapiens', descriptions, counts)  
           human_contigs = human+homo_sapiens+h_sapiens  
           human_counts = humanc+homo_sapiensc+h_sapiensc  
           pangolin, pangolinc = get_desc_count('manis javanica', descriptions,␣  
→counts)  
           pangolin_p, pangolin_pc = get_desc_count('manis pentadactyla',␣  
→descriptions, counts)  
           pangolin=pangolin+pangolin_p  
           pangolin_counts=pangolinc+pangolin_pc  
           mouse,mousec = get_desc_count('mus musculus', descriptions, counts)
```

```

vector,vectorc= get_desc_count('vector', descriptions, counts)
synthetic,syntheticc= get_desc_count('synthetic', descriptions, counts)
pig,pigc = get_desc_count('sus scrofa', descriptions, counts)
cat,catc = get_desc_count('felis catus', descriptions, counts)
tiger,tigerc = get_desc_count('panthera tigris', descriptions, counts)
dog,dogc = get_desc_count('canis lupus', descriptions, counts)
virus,virusc = get_desc_count('virus', descriptions, counts)

mulatta,mulattac = get_desc_count('mulatta', descriptions, counts)
troglodytes,troglodytesc = get_desc_count('troglodytes', descriptions,
↪counts)
pongo,pongoc =get_desc_count('pongo', descriptions, counts)
papio,papioc = get_desc_count('papio', descriptions, counts)
mandrillus,mandrillusc =get_desc_count('mandrillus', descriptions,
↪counts)
cercocebus,cercocebusc =get_desc_count('cercocebus', descriptions,
↪counts)
gelada,geladac =get_desc_count('gelada', descriptions, counts)
monkey = mulatta+troglodytes+pongo+papio+mandrillus+cercocebus+gelada
monkey_counts =
↪mulattac+troglodytesc+pongoc+papioc+mandrillusc+cercocebusc+geladac

mustela,mustelac= get_desc_count('mustela', descriptions, counts)

pipistrellus,pipistrellusc =get_desc_count('pipistrellus',
↪descriptions, counts)
rhinolophus,rhinolophusc = get_desc_count('rhinolophus', descriptions,
↪counts)
pteropus,pteropusc = get_desc_count('pteropus', descriptions, counts)
myotis,myotisc = get_desc_count('myotis', descriptions, counts)
bat = pipistrellus + rhinolophus+pteropus+myotis
bat_counts = pipistrellusc + rhinolophusc+pteropusc+myotisc

mycoplasma,mycoplasmac = get_desc_count('mycoplasma', descriptions,
↪counts)

lst = ['human', 'monkey', 'pangolin', 'mouse',
      'pig', 'cat', 'tiger', 'dog', 'bat', 'virus',
↪'vector','mycoplasma','mustela','synthetic']
lengths = [human_counts, monkey_counts, pangolin_counts, mousec,
          pigc, catc, tigerc, dogc, bat_counts, virusc, vectorc,
↪mycoplasmac, mustelac, syntheticc]
final_contigs=BASE_PATH+sra+'/megahit_default/final.contigs.fa'
with open(final_contigs) as final_contigs_file:
    total_contigs=sum(1 for _ in final_contigs_file)
res = [int(i) for i in counts]

```

```

        total_contigs_matched=sum(res)
        sra_l=[sra]*len(lst)
        fractions_matched = [human_counts/total_contigs_matched, monkey_counts/
↪total_contigs_matched, pangolin_counts/total_contigs_matched, mousec/
↪total_contigs_matched,
            pigc/total_contigs_matched, catc/total_contigs_matched, tigerc/
↪total_contigs_matched, dogc/total_contigs_matched, bat_counts/
↪total_contigs_matched,
            virusc/total_contigs_matched, vectorc/
↪total_contigs_matched, mycoplasmac/total_contigs_matched, mustelac/
↪total_contigs_matched,
            syntheticc/total_contigs_matched]
        pct_matched = [round(i * 100,2) for i in fractions_matched]
        df = pd.DataFrame(list(zip(sra_l, lst, lengths, pct_matched)),
            columns=['SRA', 'Name', 'count', 'pct_matched'])

    return df
return None

```

```

[12]: sns.set(rc={"figure.figsize":(4, 4)})
def plot_df(df, sra):
    ax=df.plot(x='Name', y='count', kind='bar')
    ax.set_title(sra, fontsize=12)
    #ax.set_yscale('log')
    #ax.set_ylim([0,df['count'].max()+10])
    ax.set_ylim(bottom=0)
    fig = plt.gcf()
    fig.savefig(BASE_PATH+sra+'/magic_blast/
↪'+f'{sra}_{dbname}_{kmer}_magicBLAST_species.png', bbox_inches="tight")

```

```

[13]: for sra in sra_list:
        print(sra)
        try:
            df=get_descr(sra)
            print(df.head())
            df.to_csv(BASE_PATH+sra+'/magic_blast/
↪'+f'{sra}_{dbname}_{kmer}_magicBLAST_species_df.csv')
            plot_df(df, sra)
        except FileNotFoundError:
            print(e)
            pass
        except AttributeError:
            pass

```

SRR11093265

sra: SRR11093265, accessions: 10, descriptions: 10, counts: 10, total: 10

	SRA	Name	count	pct_matched
0	SRR11093265	human	0	0.0

1	SRR11093265	monkey	0	0.0
2	SRR11093265	pangolin	0	0.0
3	SRR11093265	mouse	0	0.0
4	SRR11093265	pig	0	0.0

SRR11093266

sra: SRR11093266, accessions: 34, descriptions: 34, counts: 34, total: 34

	SRA	Name	count	pct_matched
0	SRR11093266	human	6	1.09
1	SRR11093266	monkey	0	0.00
2	SRR11093266	pangolin	10	1.82
3	SRR11093266	mouse	2	0.36
4	SRR11093266	pig	0	0.00

SRR11093267

sra: SRR11093267, accessions: 100, descriptions: 100, counts: 100, total: 100

	SRA	Name	count	pct_matched
0	SRR11093267	human	6	0.29
1	SRR11093267	monkey	4	0.19
2	SRR11093267	pangolin	57	2.77
3	SRR11093267	mouse	0	0.00
4	SRR11093267	pig	0	0.00

SRR11093268

sra: SRR11093268, accessions: 10, descriptions: 10, counts: 10, total: 10

	SRA	Name	count	pct_matched
0	SRR11093268	human	0	0.00
1	SRR11093268	monkey	0	0.00
2	SRR11093268	pangolin	2	1.06
3	SRR11093268	mouse	0	0.00
4	SRR11093268	pig	0	0.00

SRR11093269

sra: SRR11093269, accessions: 223, descriptions: 223, counts: 223, total: 223

	SRA	Name	count	pct_matched
0	SRR11093269	human	13	0.77
1	SRR11093269	monkey	2	0.12
2	SRR11093269	pangolin	160	9.42
3	SRR11093269	mouse	2	0.12
4	SRR11093269	pig	1	0.06

SRR11093270

sra: SRR11093270, accessions: 11832, descriptions: 11832, counts: 11832, total: 11832

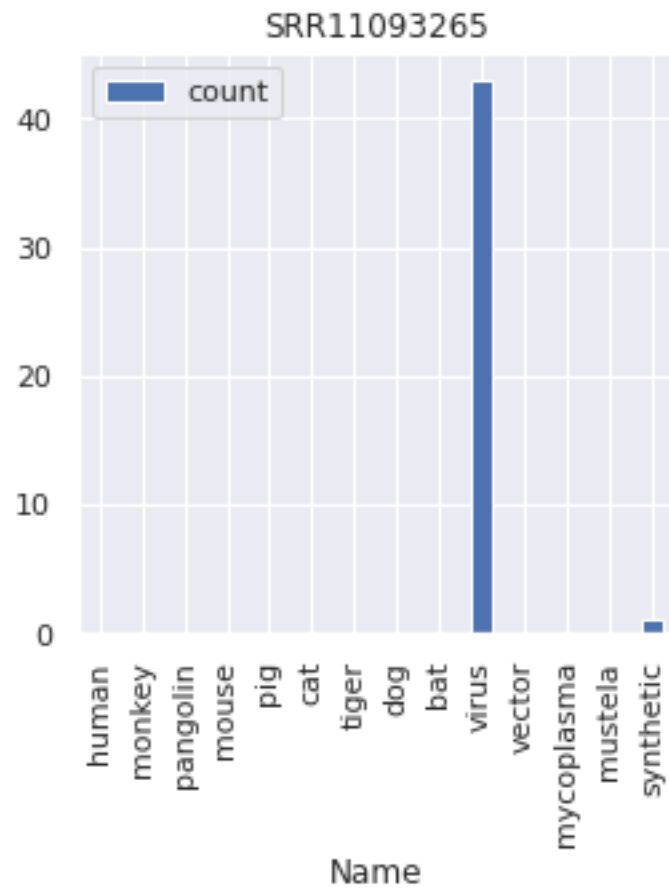
	SRA	Name	count	pct_matched
0	SRR11093270	human	110	0.58
1	SRR11093270	monkey	13	0.07
2	SRR11093270	pangolin	18413	96.57
3	SRR11093270	mouse	26	0.14
4	SRR11093270	pig	14	0.07

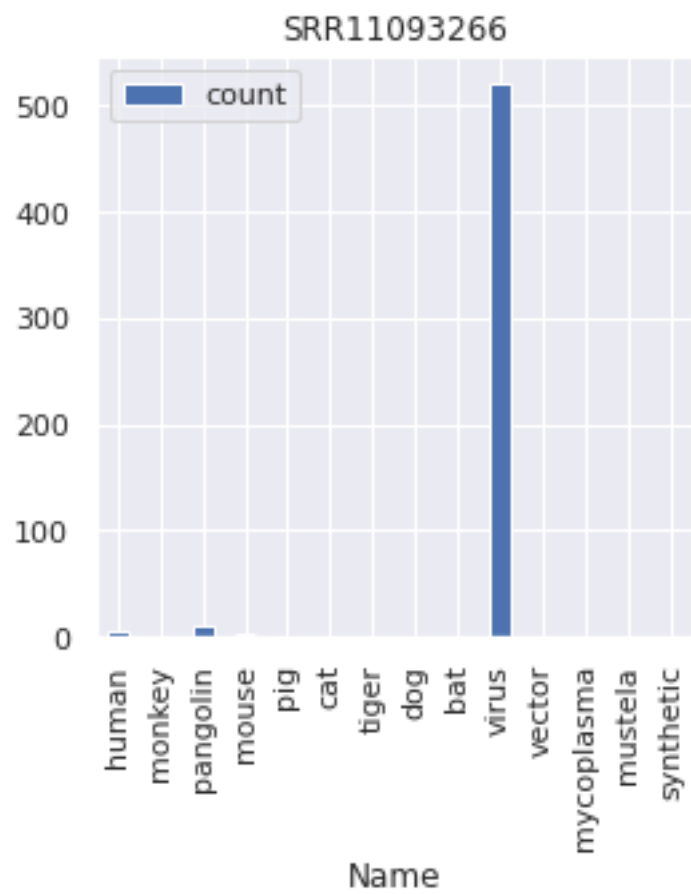
SRR11093271

sra: SRR11093271, accessions: 74, descriptions: 74, counts: 74, total: 74

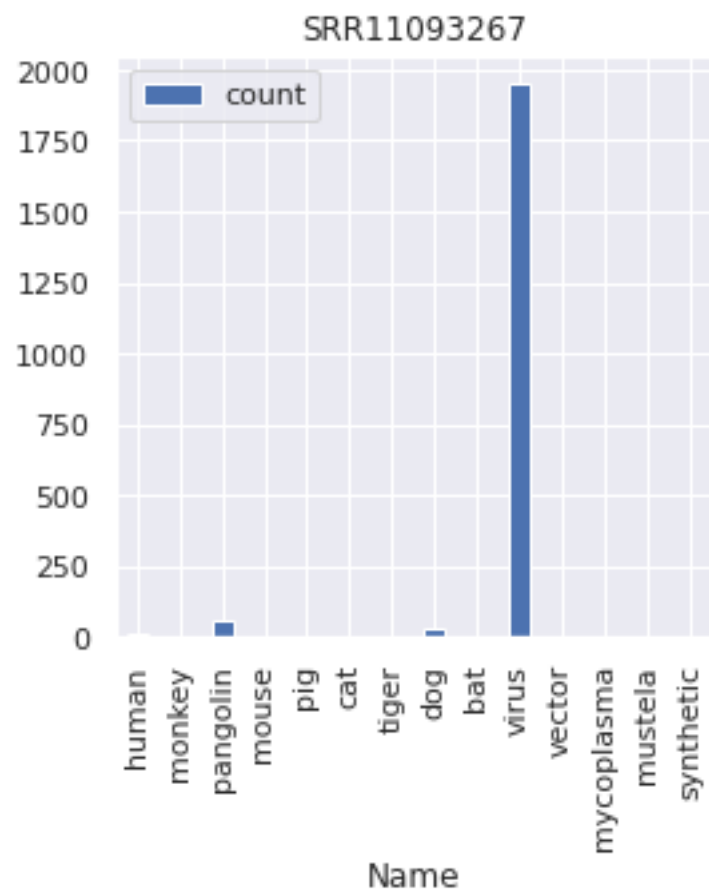
	SRA	Name	count	pct_matched
--	-----	------	-------	-------------

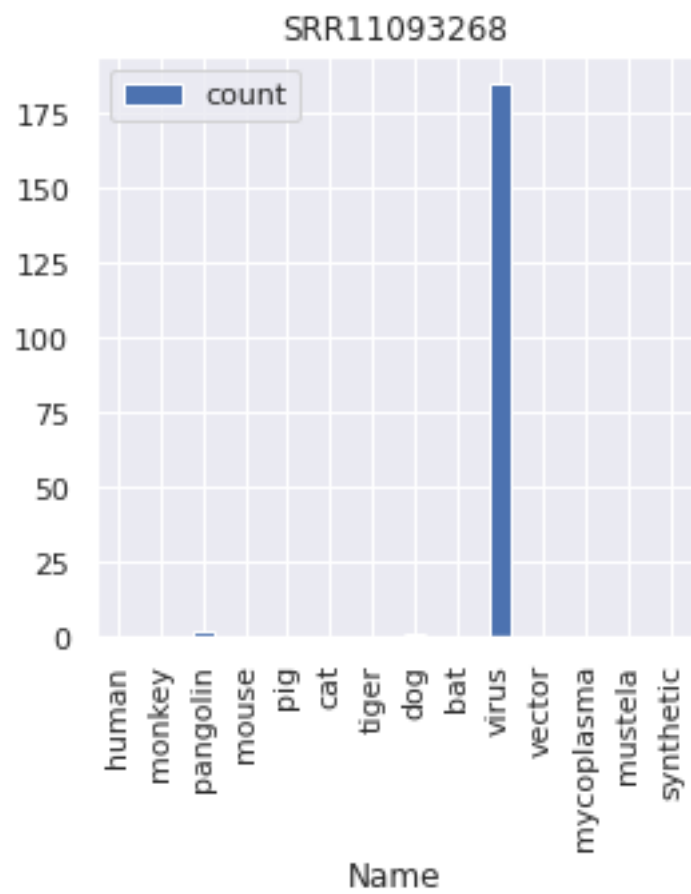
0	SRR11093271	human	4	2.56
1	SRR11093271	monkey	22	14.10
2	SRR11093271	pangolin	0	0.00
3	SRR11093271	mouse	0	0.00
4	SRR11093271	pig	0	0.00

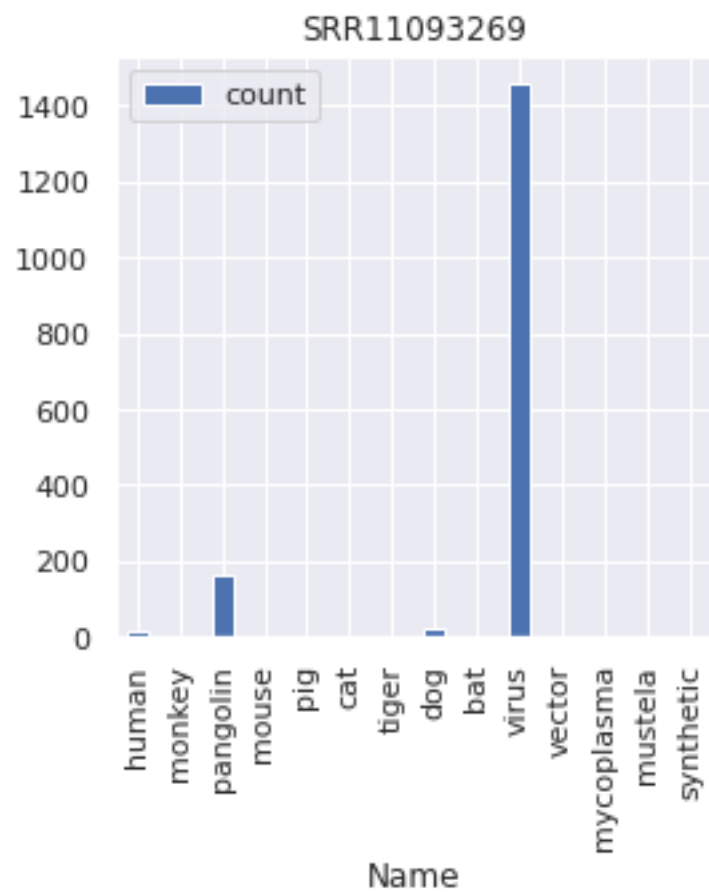


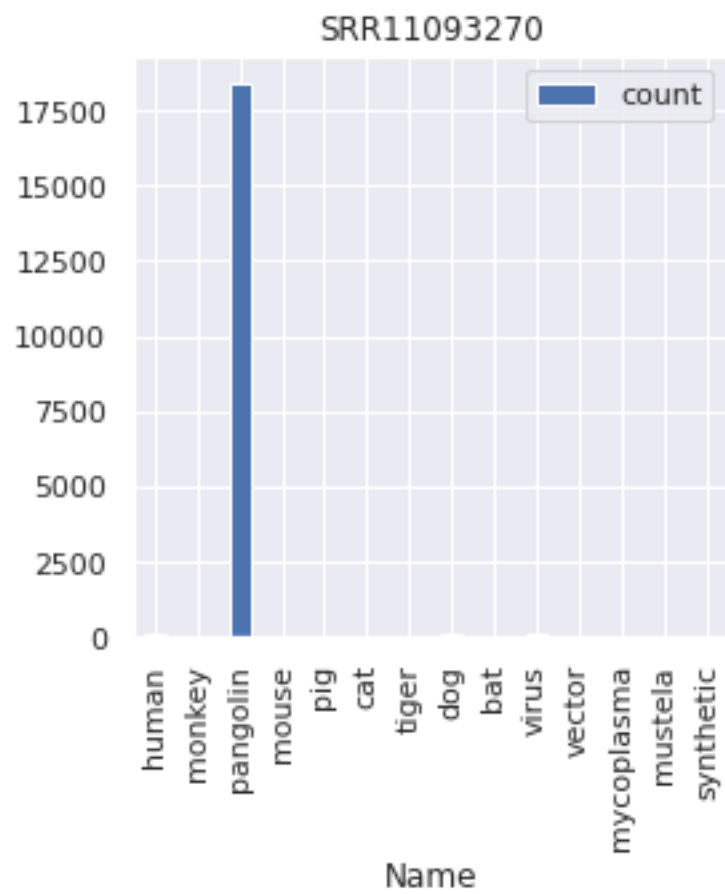


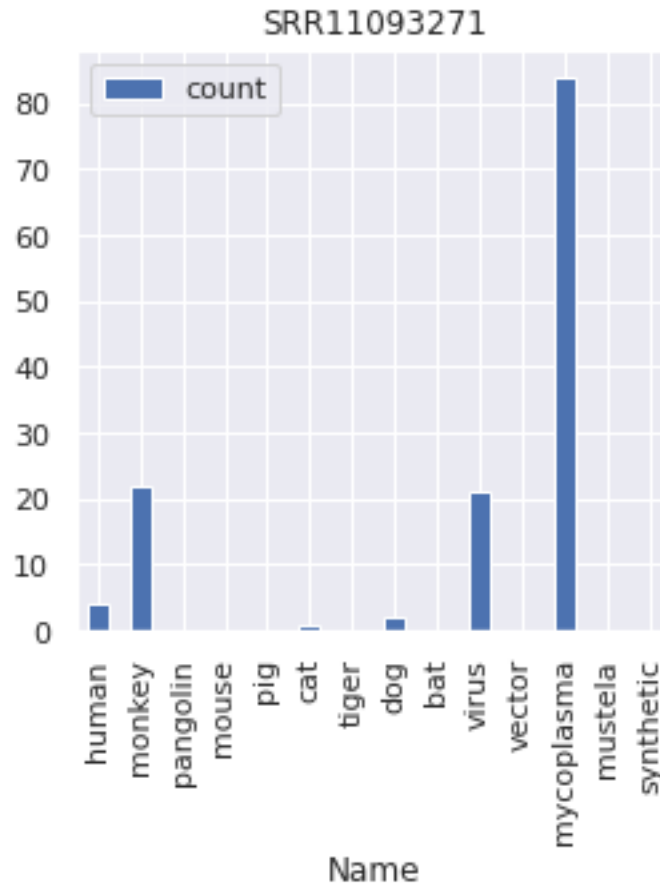












```
[14]: pathlib.Path(BASE_PATH+'general_plots/').mkdir(exist_ok=True)
```

```
def multi_plot():
    fig, axis = plt.subplots(2, 5, figsize=(16,8))
    fig.suptitle('Contig counts')
    sra=sra_list[0]
    df=get_descr(sra)
    axis[0,0].bar(df['Name'],df['count'])
    axis[0,0].tick_params(axis='x', rotation=90)
    axis[0,0].set_ylabel('count')
    axis[0,0].title.set_text(sra)

    sra=sra_list[1]
    df=get_descr(sra)
    axis[0,1].bar(df['Name'],df['count'])
    axis[0,1].set_ylabel('count')
    axis[0,1].tick_params(axis='x', rotation=90)
    axis[0,1].set_ylabel('count')
    axis[0,1].title.set_text(sra)
```

```

sra=sra_list[2]
df=get_descr(sra)
axis[0,2].bar(df['Name'],df['count'])
axis[0,2].set_ylabel('count')
axis[0,2].tick_params(axis='x', rotation=90)
axis[0,2].set_ylabel('count')
axis[0,2].title.set_text(sra)

sra=sra_list[3]
df=get_descr(sra)
axis[0,3].bar(df['Name'],df['count'])
axis[0,3].set_ylabel('count')
axis[0,3].tick_params(axis='x', rotation=90)
axis[0,3].set_ylabel('count')
axis[0,3].title.set_text(sra)

sra=sra_list[4]
df=get_descr(sra)
axis[0,4].bar(df['Name'],df['count'])
axis[0,4].set_ylabel('count')
axis[0,4].tick_params(axis='x', rotation=90)
axis[0,4].set_ylabel('count')
axis[0,4].title.set_text(sra)

sra=sra_list[5]
df=get_descr(sra)
axis[1,0].bar(df['Name'],df['count'])
axis[1,0].set_ylabel('count')
axis[1,0].tick_params(axis='x', rotation=90)
axis[1,0].set_ylabel('count')
axis[1,0].title.set_text(sra)

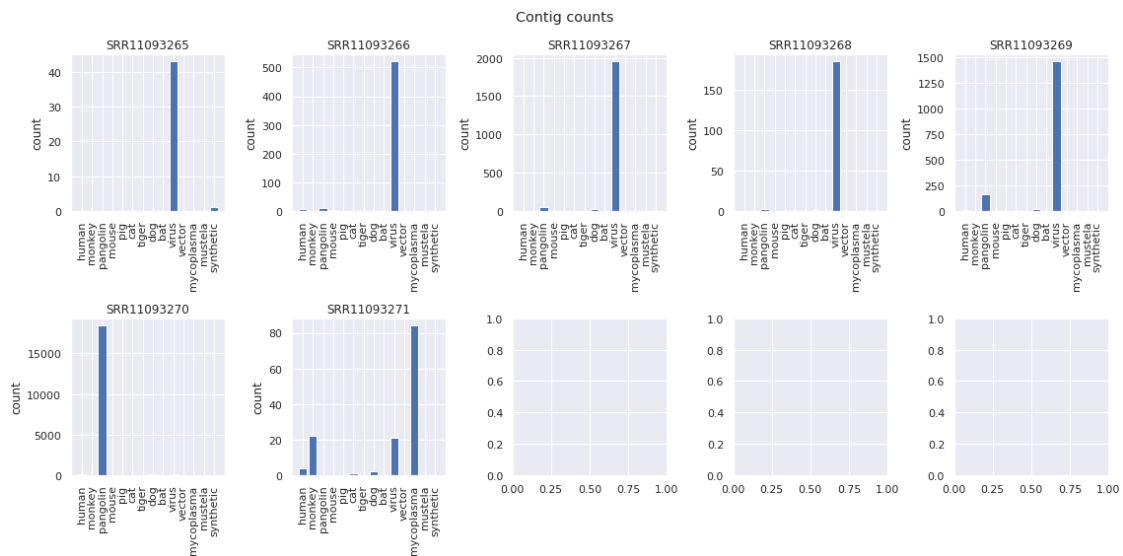
sra=sra_list[6]
df=get_descr(sra)
axis[1,1].bar(df['Name'],df['count'])
axis[1,1].set_ylabel('count')
axis[1,1].tick_params(axis='x', rotation=90)
axis[1,1].set_ylabel('count')
axis[1,1].title.set_text(sra)

fig.tight_layout()
fig.savefig(BASE_PATH+'general_plots/
↳'+f'{sra}_{dbname}_{kmer}_magicBLAST_contig_count_matrix.png',
↳bbox_inches="tight")
plt.show()

```

```
[15]: multi_plot()
```

```
sra: SRR11093265, accessions: 10, descriptions: 10, counts: 10, total: 10
sra: SRR11093266, accessions: 34, descriptions: 34, counts: 34, total: 34
sra: SRR11093267, accessions: 100, descriptions: 100, counts: 100, total: 100
sra: SRR11093268, accessions: 10, descriptions: 10, counts: 10, total: 10
sra: SRR11093269, accessions: 223, descriptions: 223, counts: 223, total: 223
sra: SRR11093270, accessions: 11832, descriptions: 11832, counts: 11832, total: 11832
sra: SRR11093271, accessions: 74, descriptions: 74, counts: 74, total: 74
```



```
[16]: frames=[]
for i,sra in enumerate(sra_list):
    try:
        df = pd.read_csv(BASE_PATH+sra+'/magic_blast/
        ↳'+f'{sra}_{dbname}_{kmer}_magicBLAST_species_df.csv')
        x = df.Name.astype('category')
        df['species_uid'] =x.cat.codes
        df['SRA_val'] =df.SRA.str.strip('SRR')
        df['SRA_val'] = pd.to_numeric(df['SRA_val'])
        frames.append(df)
    except FileNotFoundError:
        pass
df_sra = pd.concat(frames)
```

```
[17]: df_sra.drop(columns=['Unnamed: 0'],inplace=True)
```

```
[18]: df_sra.head(n=10)
```

```
[18]:
```

	SRA	Name	count	pct_matched	species_uid	SRA_val
0	SRR11093265	human	0	0.00	3	11093265
1	SRR11093265	monkey	0	0.00	4	11093265
2	SRR11093265	pangolin	0	0.00	8	11093265
3	SRR11093265	mouse	0	0.00	5	11093265
4	SRR11093265	pig	0	0.00	9	11093265
5	SRR11093265	cat	0	0.00	1	11093265
6	SRR11093265	tiger	0	0.00	11	11093265
7	SRR11093265	dog	0	0.00	2	11093265
8	SRR11093265	bat	0	0.00	0	11093265
9	SRR11093265	virus	43	97.73	13	11093265

```
[19]: df_sra.Name.unique()
```

```
[19]: array(['human', 'monkey', 'pangolin', 'mouse', 'pig', 'cat', 'tiger',  
        'dog', 'bat', 'virus', 'vector', 'mycoplasma', 'mustela',  
        'synthetic'], dtype=object)
```

### 0.0.3 All nt database matches

```
[20]: total_dict={}  
asc_desc={}  
for sra in sra_list:  
    accessions, descriptions, counts, total=get_asc_descr_count(sra)  
    for asc,desc,cnt in zip(accessions,descriptions, counts):  
        if asc in total_dict:  
            total_dict[asc]+=int(cnt)  
        else:  
            total_dict[asc]=int(cnt)  
        if asc not in asc_desc:  
            asc_desc[asc]=desc
```

```
[21]: assert len(total_dict)==len(asc_desc)
```

```
[22]: len(asc_desc)
```

```
[22]: 12158
```

Print out the 100 most common nt database classification for all contigs in the project

```
[23]: def print_top_n_sp(total_dict, max_num):  
    listofTuples = sorted(total_dict.items() , reverse=True, key=lambda x: x[1])  
    for i, elem in enumerate(listofTuples):  
        if i<max_num:  
            print(asc_desc[elem[0]], ": " , elem[1] )
```



```

else:
    break

```

```
[24]: print_top_n_sp(total_dict, max_num=10)
```

```

Pangolin coronavirus isolate PCoV_GX-P5L, complete genome : 837
Pangolin coronavirus isolate PCoV_GX-P1E, complete genome : 787
Pangolin coronavirus isolate PCoV_GX-P2V, complete genome : 756
Pangolin coronavirus isolate PCoV_GX-P4L, complete genome : 715
Pangolin coronavirus isolate PCoV_GX-P5E, complete genome : 641
Pangolin coronavirus isolate PCoV_GX-P3B genomic sequence : 478
PREDICTED: Manis javanica proline rich coiled-coil 2A (PRRC2A), mRNA : 15
Pangolin coronavirus isolate MP789, complete genome : 14
Manis javanica isolate MP_PG03-UM mitochondrion, complete genome : 14
PREDICTED: Manis javanica LDL receptor related protein 1 (LRP1), transcript
variant X3, mRNA : 14

```

#### 0.0.4 Human

```
[25]: df=df_sra[df_sra['Name'].str.contains('human')]
```

```

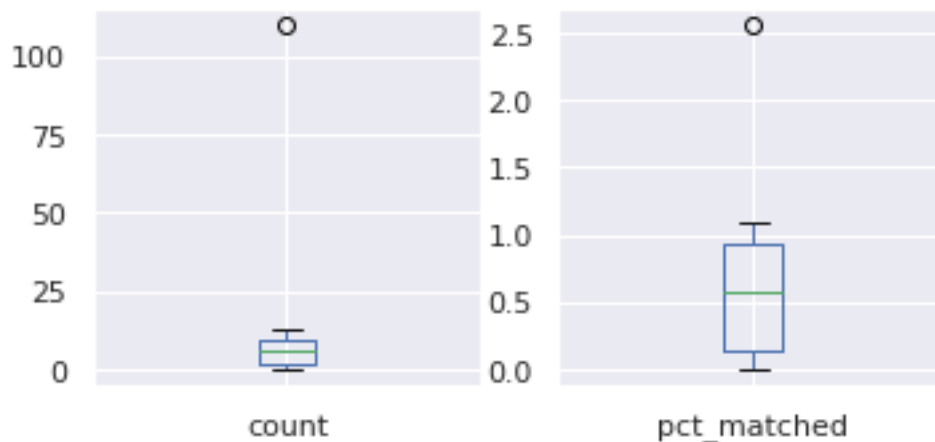
[26]: df.drop(columns=['Name'],inplace=True)
sns.set(rc={"figure.figsize":(6, 6)})
df_box=df[['count', 'pct_matched']]
df_box.plot(kind='box', subplots=True, layout=(4,4), sharex=False,
↪sharey=False, figsize=(12,12))

```

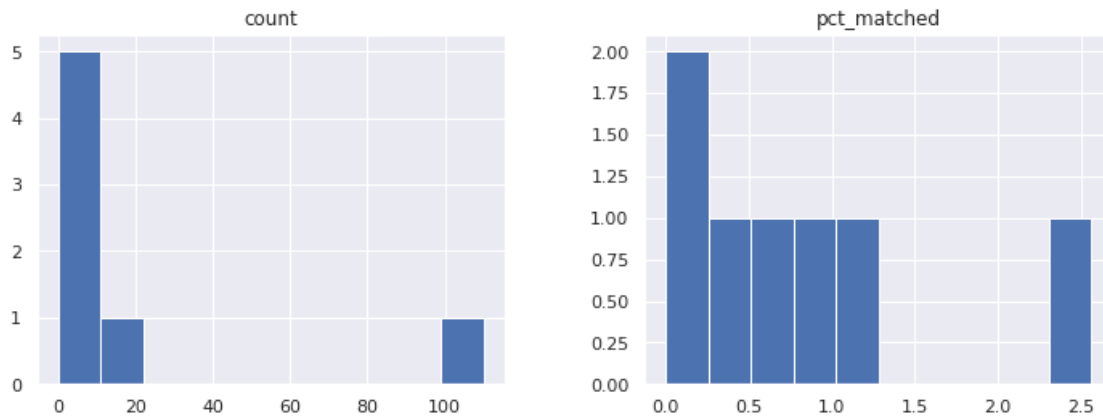
```

[26]: count          AxesSubplot(0.125,0.71587;0.168478x0.16413)
pct_matched        AxesSubplot(0.327174,0.71587;0.168478x0.16413)
dtype: object

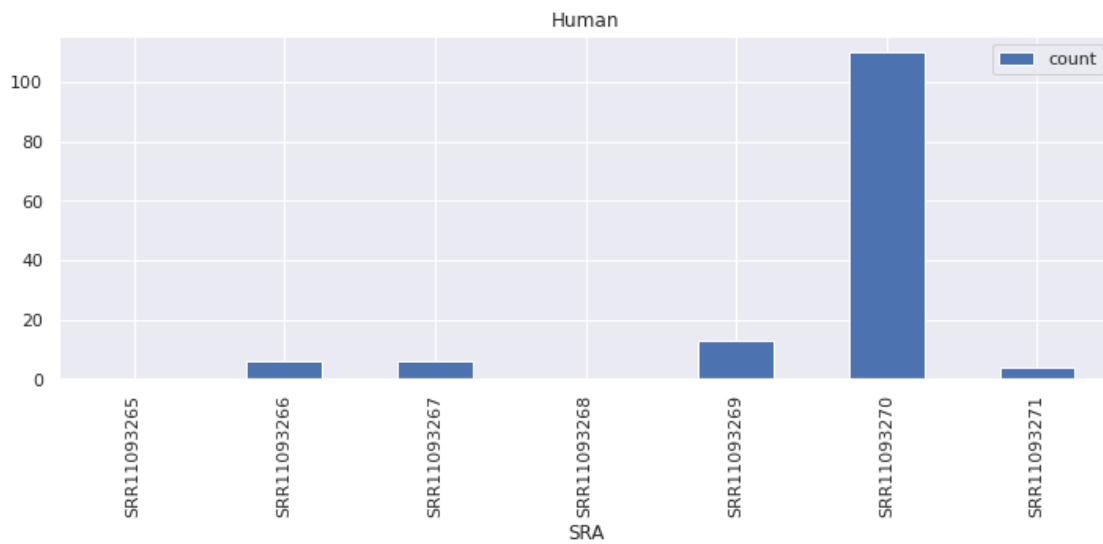
```



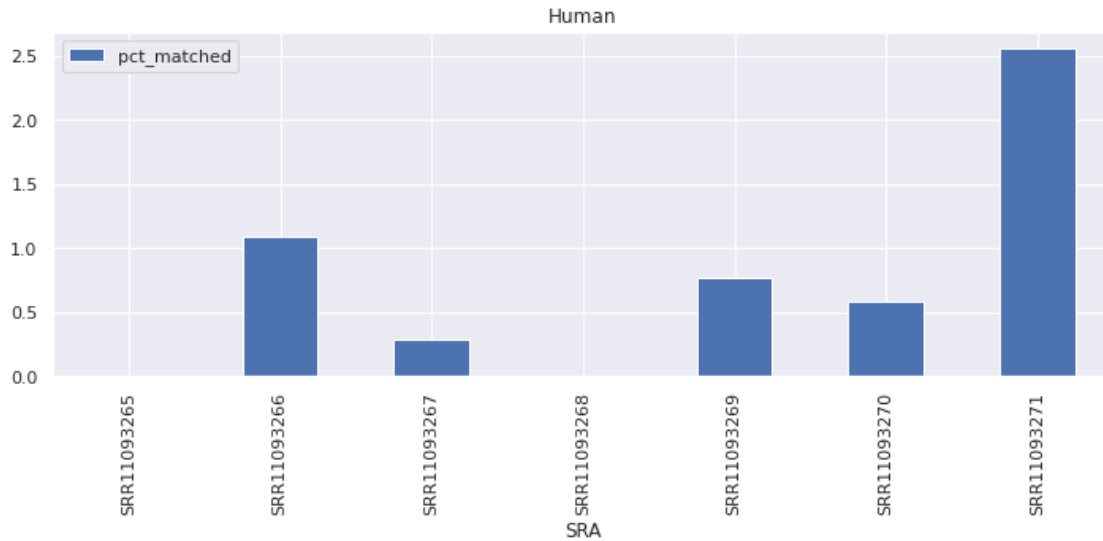
```
[27]: df_box.hist(figsize=(12,4))
plt.show()
```



```
[28]: ax=df.plot(x='SRA', y='count', kind='bar',figsize=(12,4))
ax.set_title('Human', fontsize=12)
plt.show()
```



```
[29]: ax=df.plot(x='SRA', y='pct_matched', kind='bar',figsize=(12,4))
ax.set_title('Human', fontsize=12)
plt.show()
```



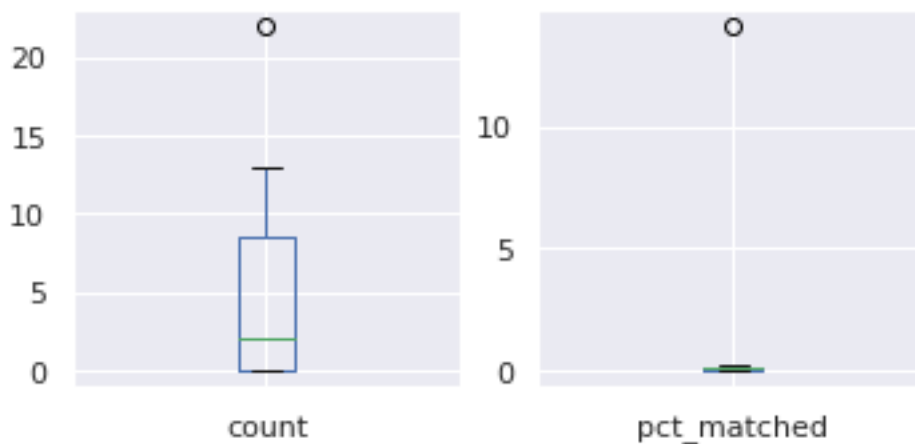
### 0.0.5 Monkey

```
[30]: s_name='monkey'
```

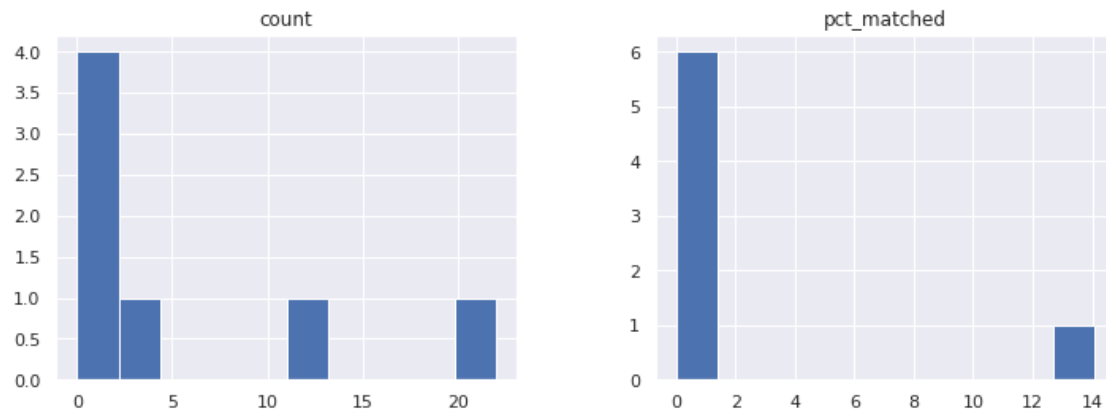
```
[31]: df=df_sra[df_sra['Name'].str.contains(s_name)]
```

```
[32]: df.drop(columns=['Name'],inplace=True)
sns.set(rc={"figure.figsize":(6, 6)})
df_box=df[['count', 'pct_matched']]
df_box.plot(kind='box', subplots=True, layout=(4,4), sharex=False,
↪sharey=False, figsize=(12,12))
```

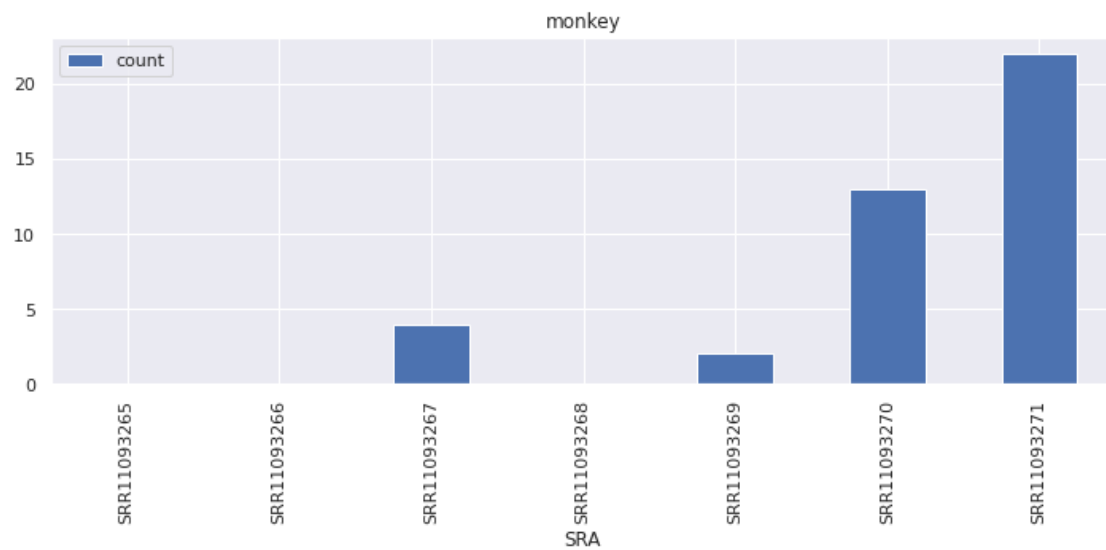
```
[32]: count          AxesSubplot(0.125,0.71587;0.168478x0.16413)
pct_matched        AxesSubplot(0.327174,0.71587;0.168478x0.16413)
dtype: object
```



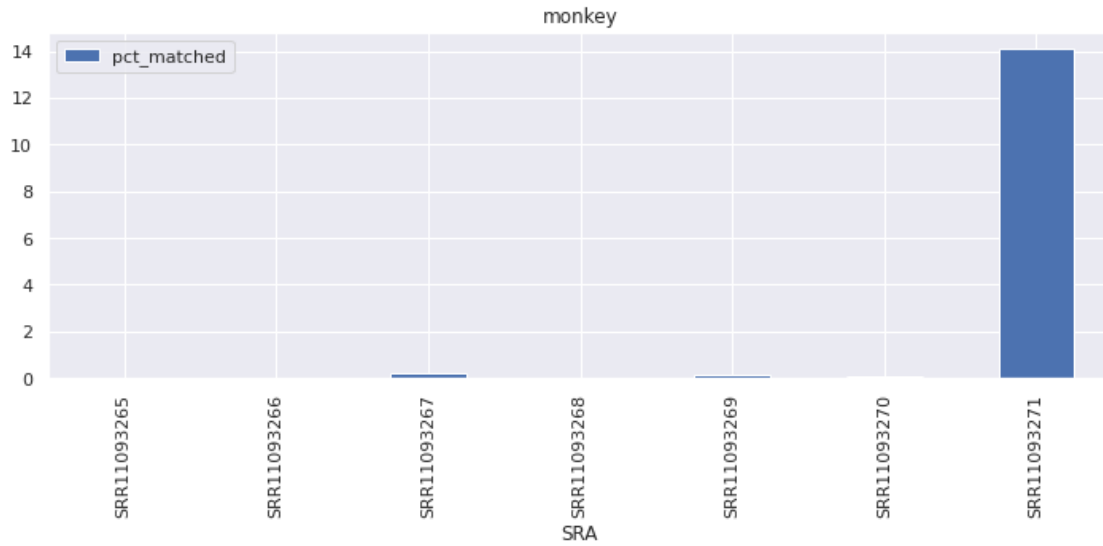
```
[33]: df_box.hist(figsize=(12,4))
plt.show()
```



```
[34]: ax=df.plot(x='SRA', y='count', kind='bar',figsize=(12,4))
ax.set_title(s_name, fontsize=12)
plt.show()
```



```
[35]: ax=df.plot(x='SRA', y='pct_matched', kind='bar',figsize=(12,4))
ax.set_title(s_name, fontsize=12)
plt.show()
```



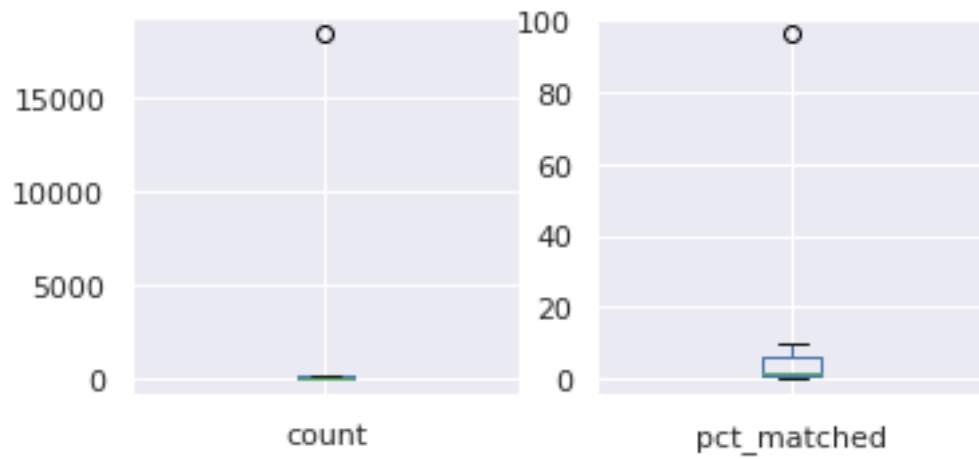
### 0.0.6 Pangolin

```
[36]: s_name='pangolin'
```

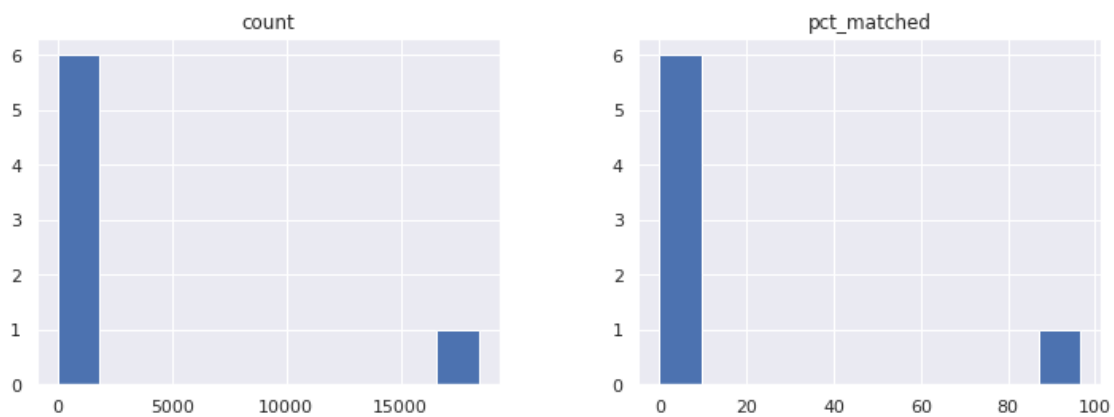
```
[37]: df=df_sra[df_sra['Name'].str.contains(s_name)]
```

```
[38]: df.drop(columns=['Name'],inplace=True)
sns.set(rc={"figure.figsize":(6, 6)})
df_box=df[['count', 'pct_matched']]
df_box.plot(kind='box', subplots=True, layout=(4,4), sharex=False,
↪sharey=False, figsize=(12,12))
```

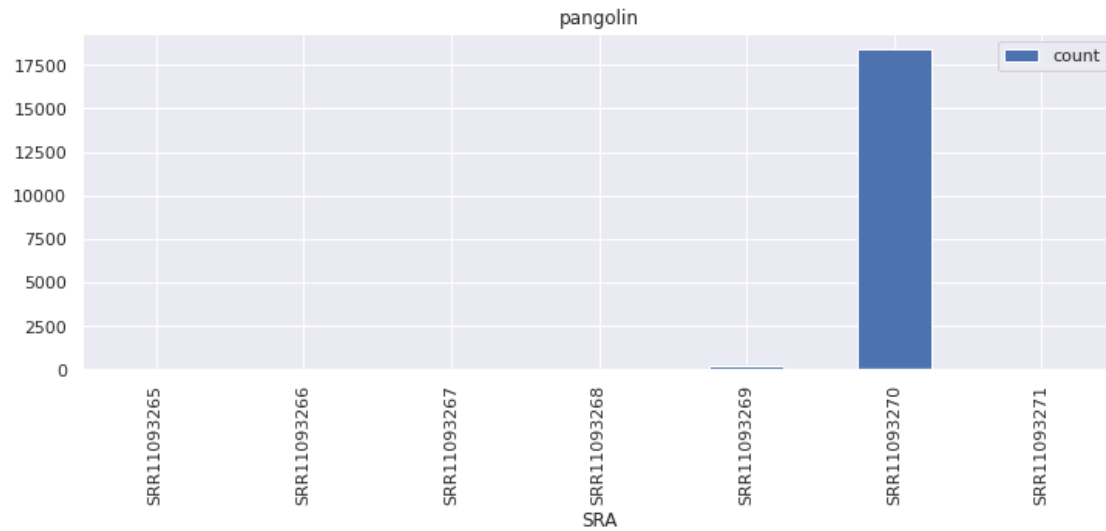
```
[38]: count          AxesSubplot(0.125,0.71587;0.168478x0.16413)
pct_matched        AxesSubplot(0.327174,0.71587;0.168478x0.16413)
dtype: object
```



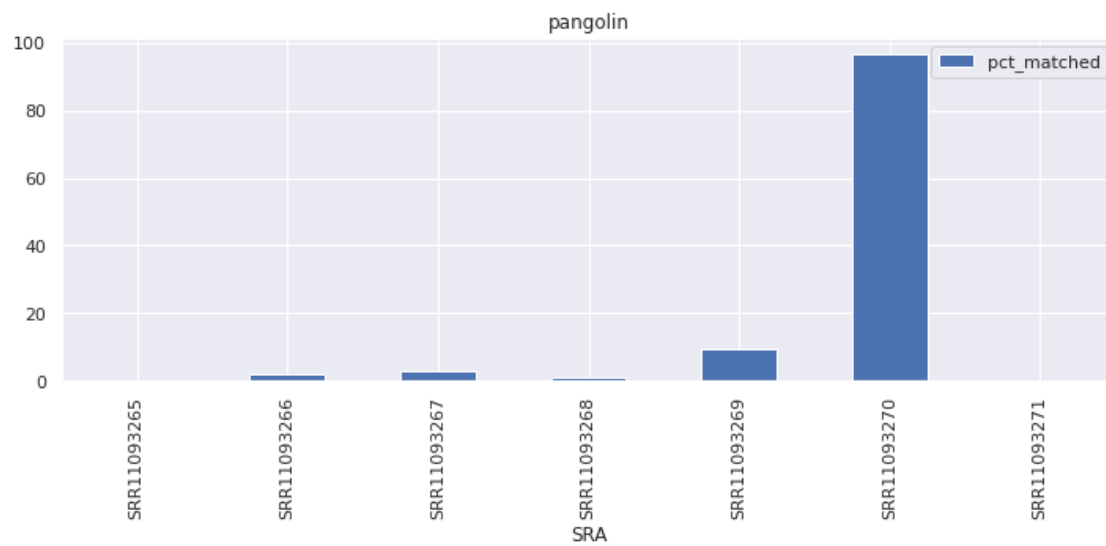
```
[39]: df_box.hist(figsize=(12,4))
plt.show()
```



```
[40]: ax=df.plot(x='SRA', y='count', kind='bar',figsize=(12,4))
ax.set_title(s_name, fontsize=12)
plt.show()
```



```
[41]: ax=df.plot(x='SRA', y='pct_matched', kind='bar',figsize=(12,4))
ax.set_title(s_name, fontsize=12)
plt.show()
```



### 0.0.7 Mouse

```
[42]: s_name='mouse'
```

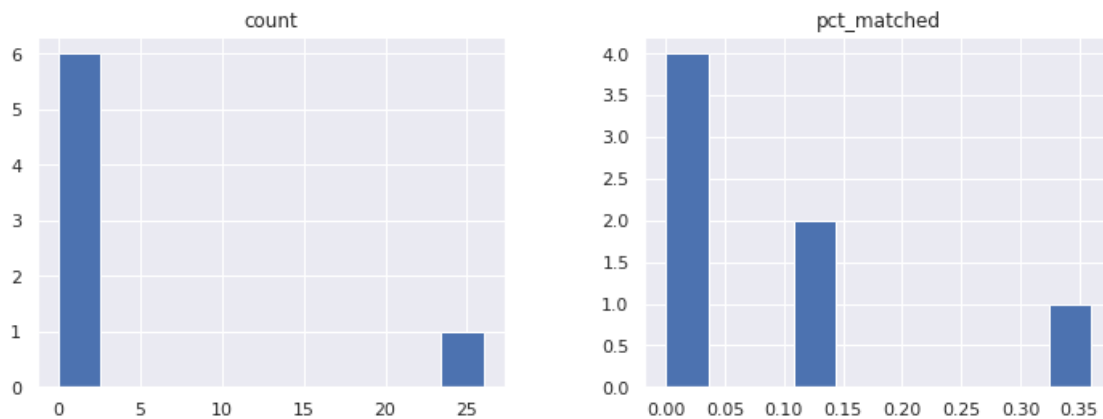
```
[43]: df=df_sra[df_sra['Name'].str.contains(s_name)]
```

```
[44]: df.drop(columns=['Name'],inplace=True)
sns.set(rc={"figure.figsize":(6, 6)})
df_box=df[['count', 'pct_matched']]
df_box.plot(kind='box', subplots=True, layout=(4,4), sharex=False,
→sharey=False, figsize=(12,12))
```

```
[44]: count          AxesSubplot(0.125,0.71587;0.168478x0.16413)
pct_matched      AxesSubplot(0.327174,0.71587;0.168478x0.16413)
dtype: object
```

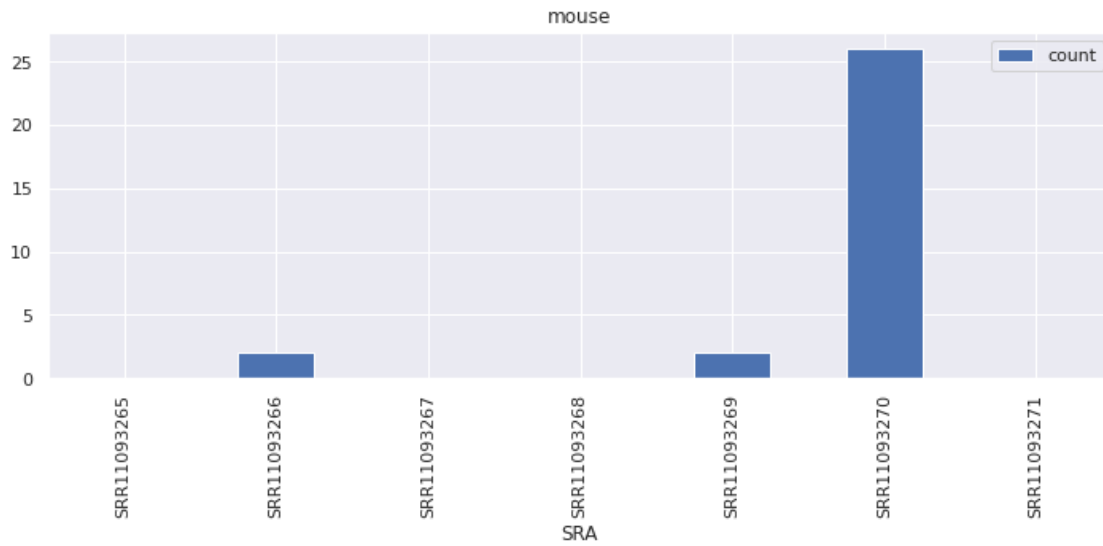


```
[45]: df_box.hist(figsize=(12,4))
plt.show()
```

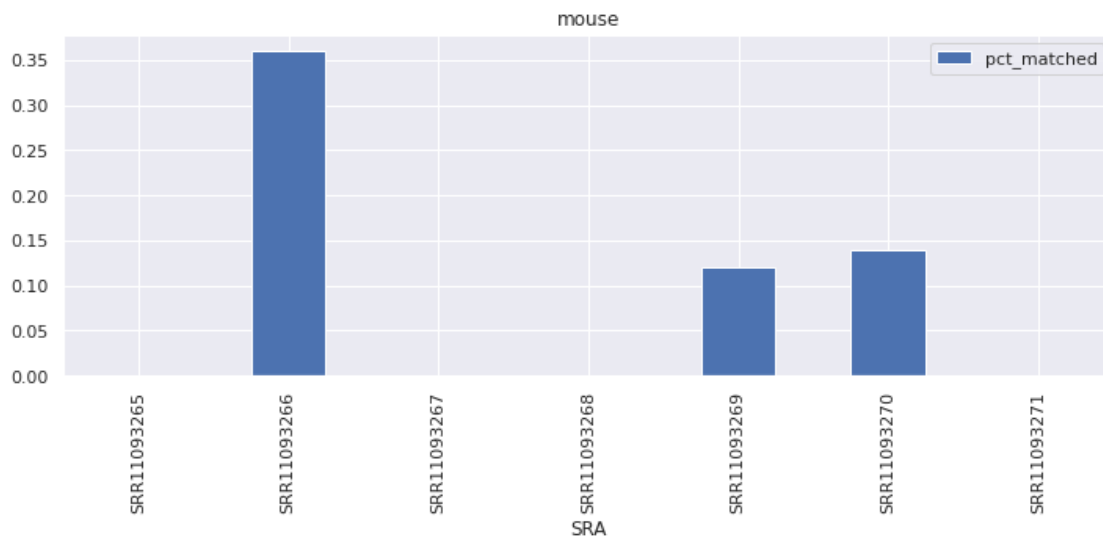


```
[46]: ax=df.plot(x='SRA', y='count', kind='bar',figsize=(12,4))
ax.set_title(s_name, fontsize=12)
plt.show()
```





```
[47]: ax=df.plot(x='SRA', y='pct_matched', kind='bar',figsize=(12,4))
      ax.set_title(s_name, fontsize=12)
      plt.show()
```



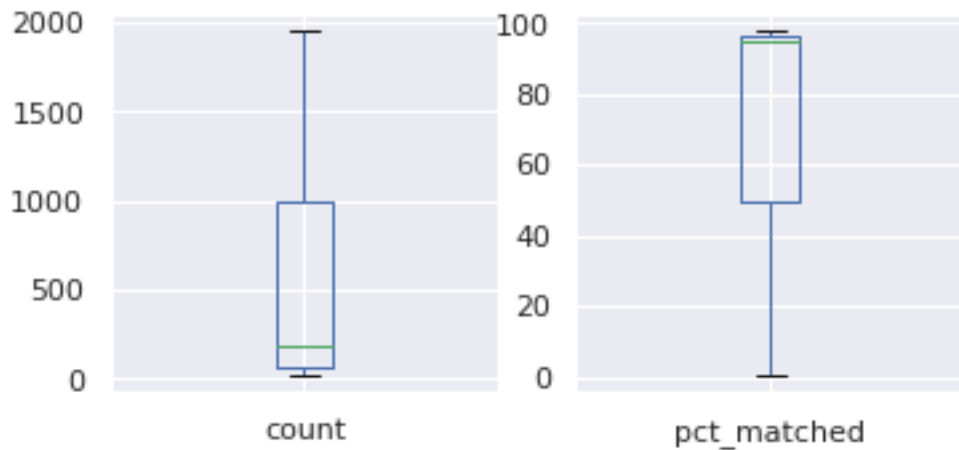
#### 0.0.8 virus

```
[48]: s_name='virus'
```

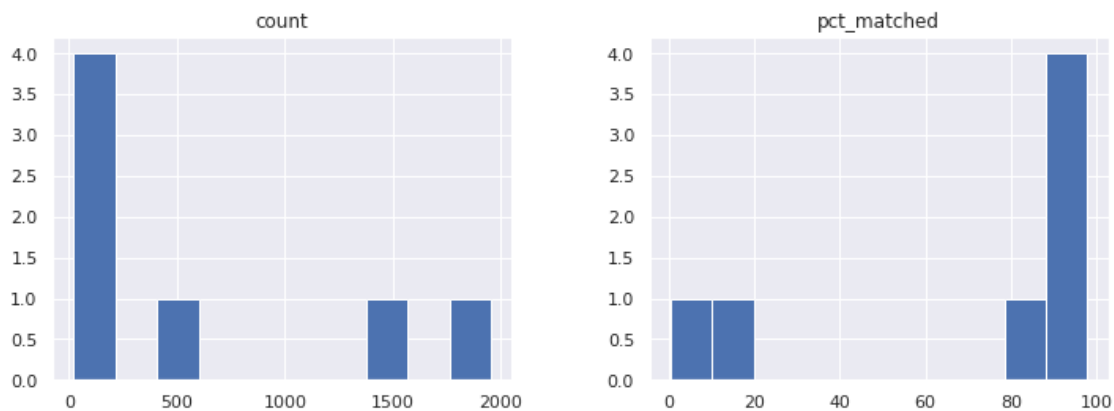
```
[49]: df=df_sra[df_sra['Name'].str.contains(s_name)]
```

```
[50]: df.drop(columns=['Name'],inplace=True)
sns.set(rc={"figure.figsize":(6, 6)})
df_box=df[['count', 'pct_matched']]
df_box.plot(kind='box', subplots=True, layout=(4,4), sharex=False,
→sharey=False, figsize=(12,12))
```

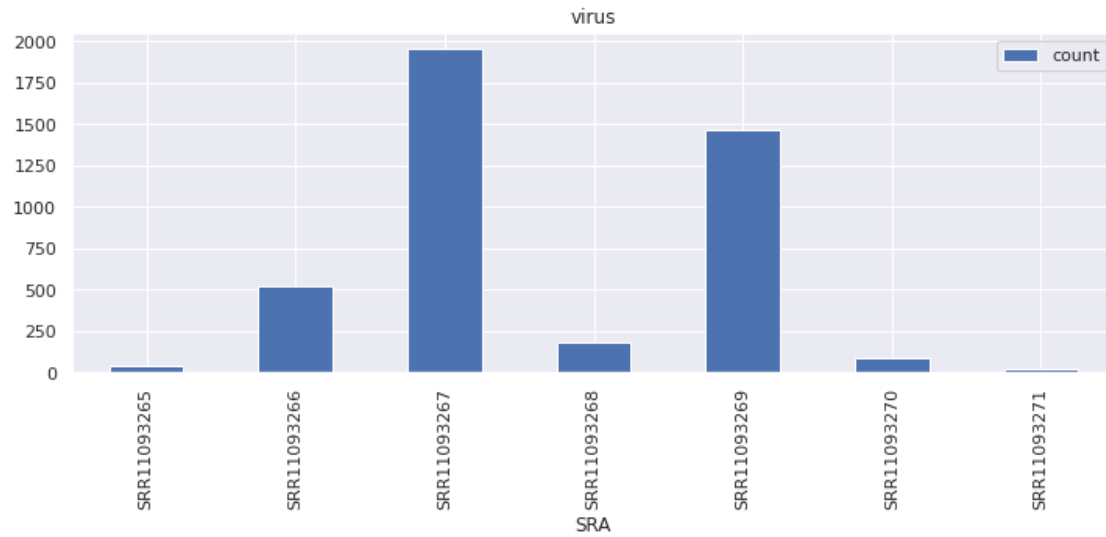
```
[50]: count          AxesSubplot(0.125,0.71587;0.168478x0.16413)
pct_matched      AxesSubplot(0.327174,0.71587;0.168478x0.16413)
dtype: object
```



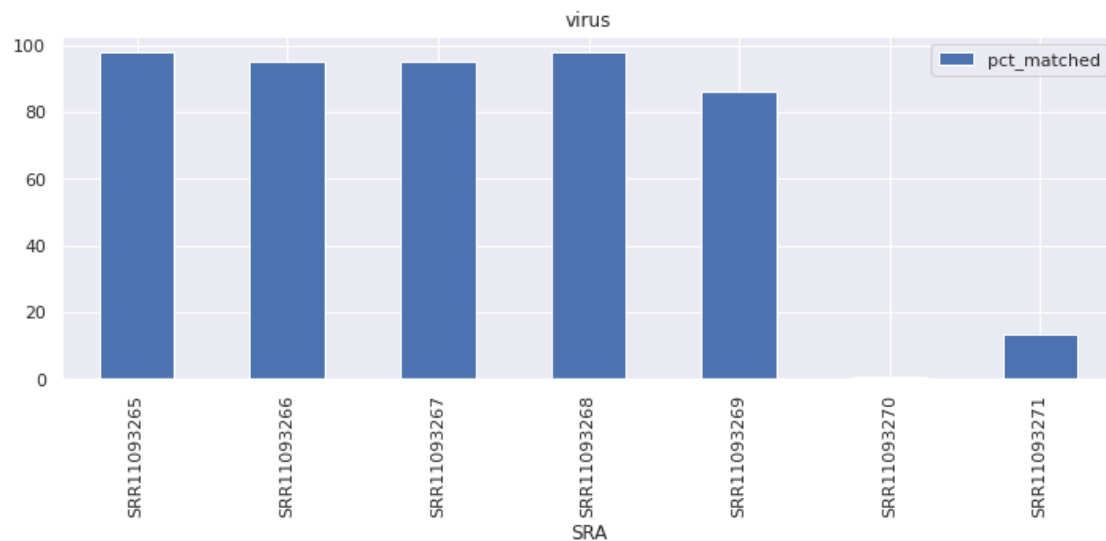
```
[51]: df_box.hist(figsize=(12,4))
plt.show()
```



```
[52]: ax=df.plot(x='SRA', y='count', kind='bar',figsize=(12,4))
ax.set_title(s_name, fontsize=12)
plt.show()
```



```
[53]: ax=df.plot(x='SRA', y='pct_matched', kind='bar',figsize=(12,4))
      ax.set_title(s_name, fontsize=12)
      plt.show()
```



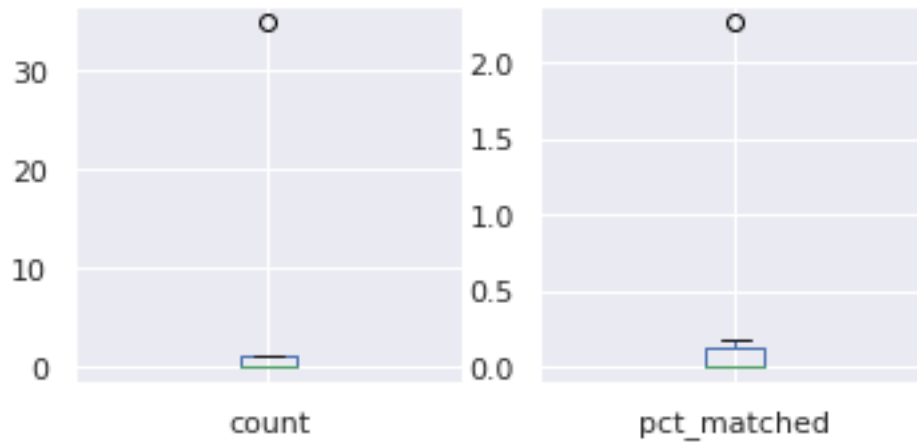
### 0.0.9 synthetic

```
[54]: s_name='synthetic'
```

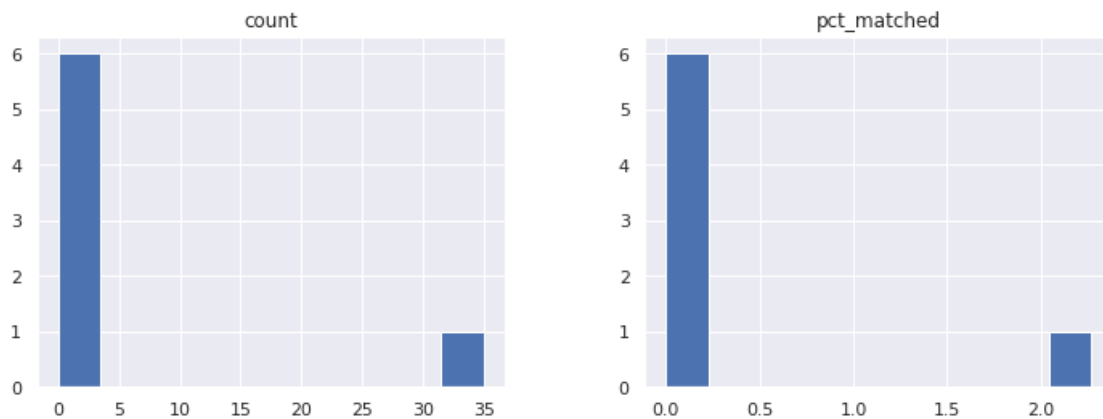
```
[55]: df=df_sra[df_sra['Name'].str.contains(s_name)]
```

```
[56]: df.drop(columns=['Name'],inplace=True)
sns.set(rc={"figure.figsize":(6, 6)})
df_box=df[['count', 'pct_matched']]
df_box.plot(kind='box', subplots=True, layout=(4,4), sharex=False,
→sharey=False, figsize=(12,12))
```

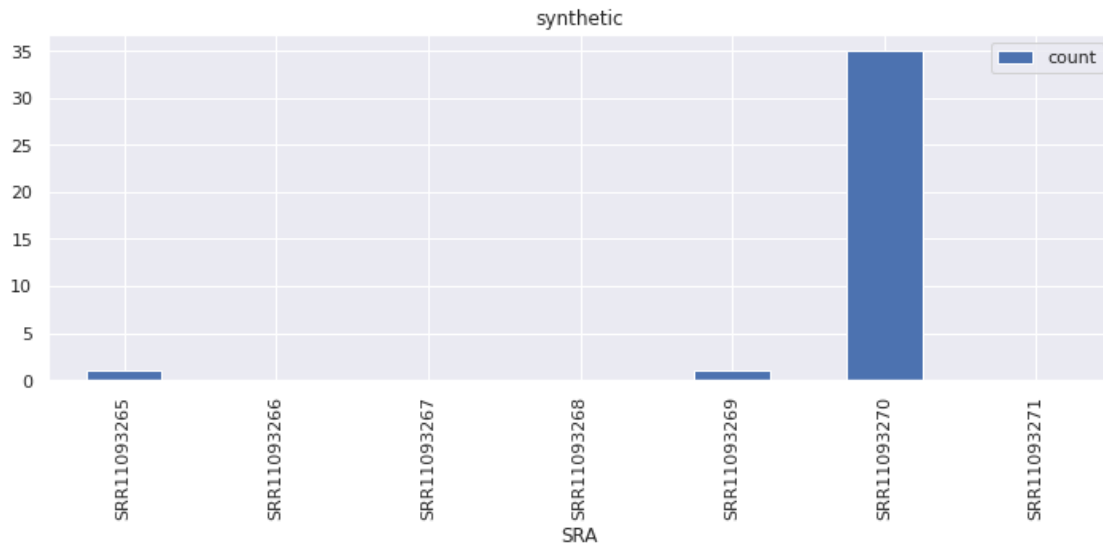
```
[56]: count          AxesSubplot(0.125,0.71587;0.168478x0.16413)
pct_matched      AxesSubplot(0.327174,0.71587;0.168478x0.16413)
dtype: object
```



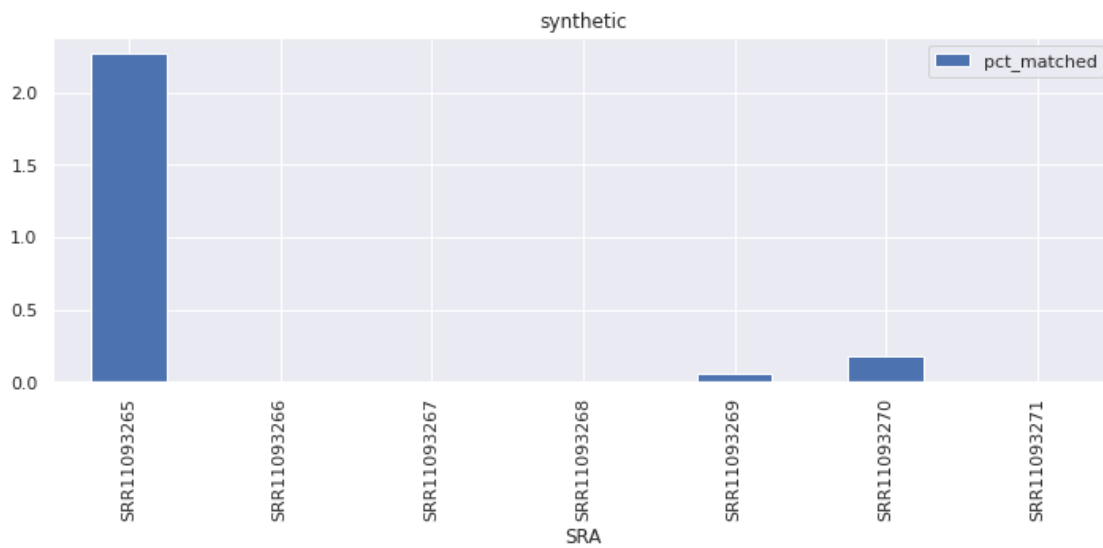
```
[57]: df_box.hist(figsize=(12,4))
plt.show()
```



```
[58]: ax=df.plot(x='SRA', y='count', kind='bar',figsize=(12,4))
ax.set_title(s_name, fontsize=12)
plt.show()
```



```
[59]: ax=df.plot(x='SRA', y='pct_matched', kind='bar',figsize=(12,4))
ax.set_title(s_name, fontsize=12)
plt.show()
```



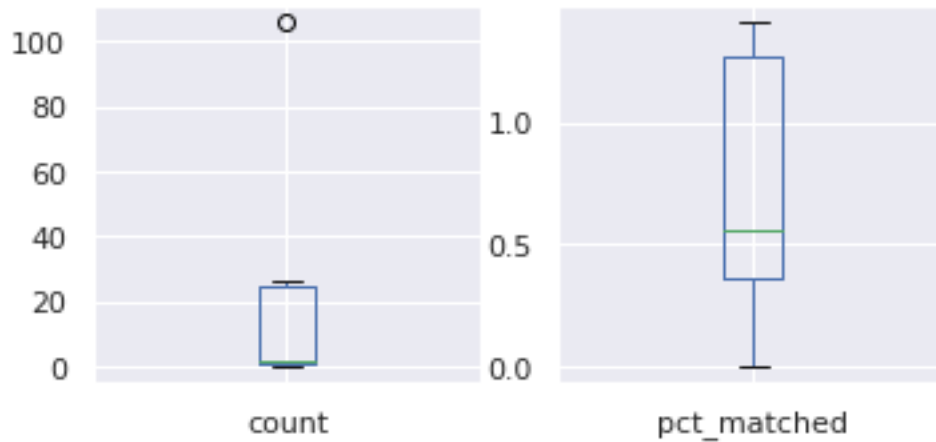
#### 0.0.10 dog

```
[60]: s_name='dog'
```

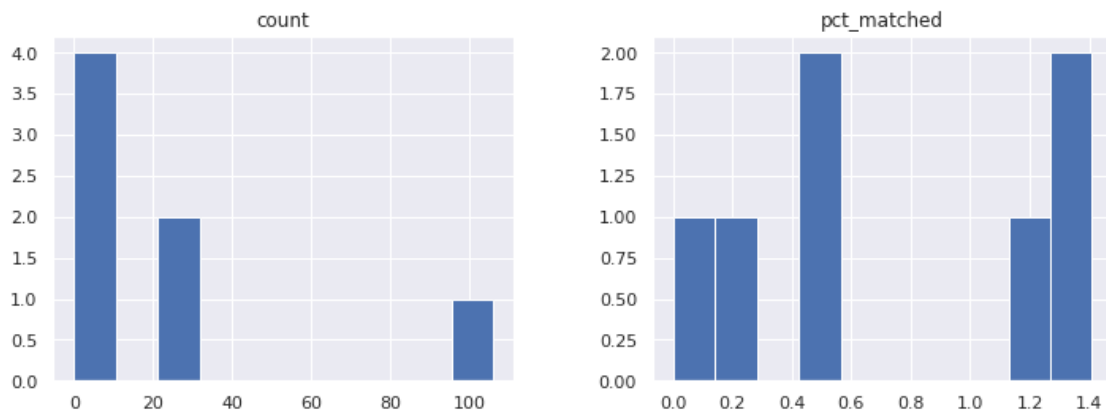
```
[61]: df=df_sra[df_sra['Name'].str.contains(s_name)]
```

```
[62]: df.drop(columns=['Name'],inplace=True)
sns.set(rc={"figure.figsize":(6, 6)})
df_box=df[['count', 'pct_matched']]
df_box.plot(kind='box', subplots=True, layout=(4,4), sharex=False,
→sharey=False, figsize=(12,12))
```

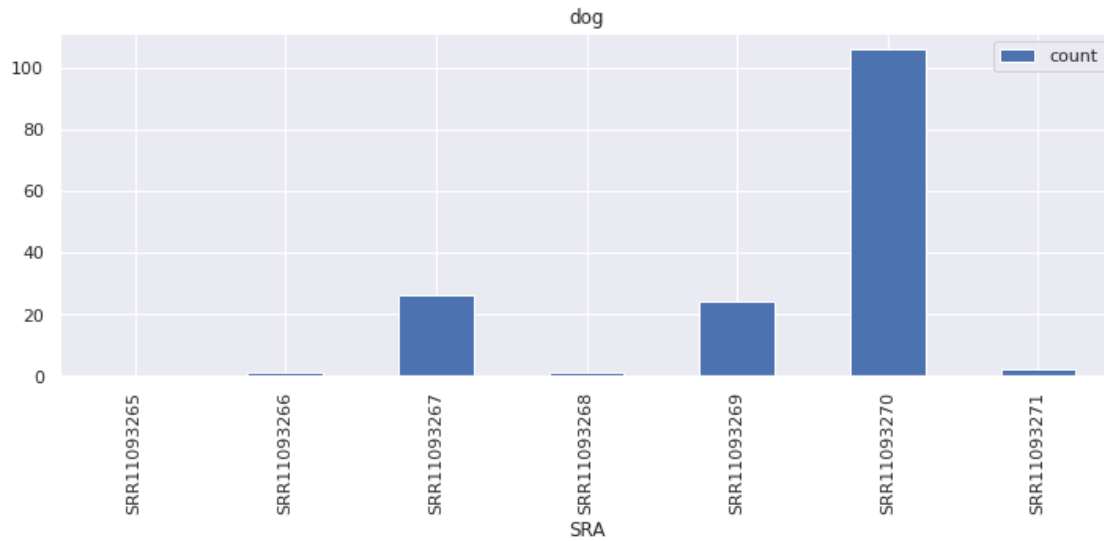
```
[62]: count          AxesSubplot(0.125,0.71587;0.168478x0.16413)
pct_matched      AxesSubplot(0.327174,0.71587;0.168478x0.16413)
dtype: object
```



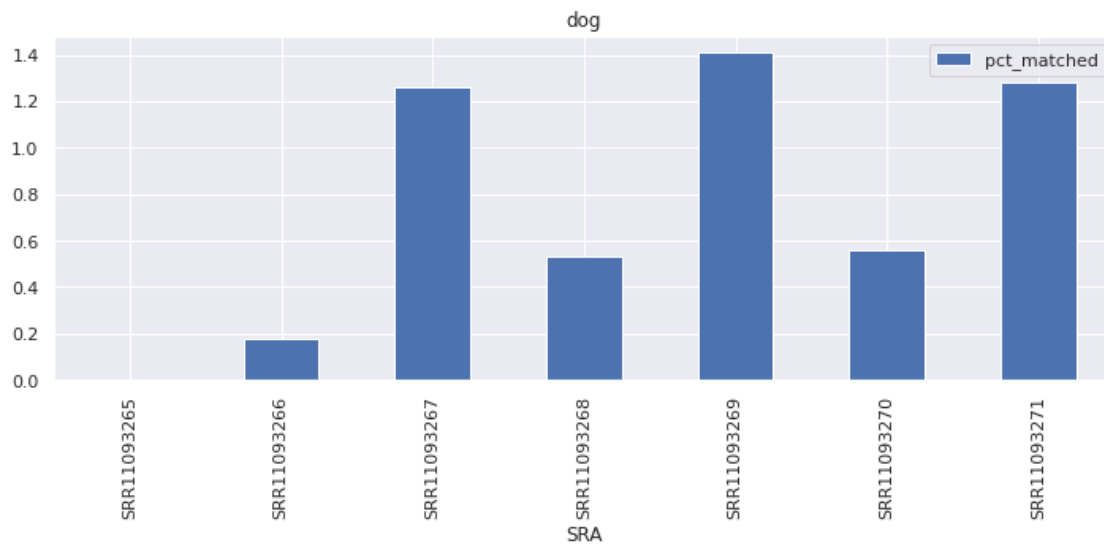
```
[63]: df_box.hist(figsize=(12,4))
plt.show()
```



```
[64]: ax=df.plot(x='SRA', y='count', kind='bar',figsize=(12,4))
ax.set_title(s_name, fontsize=12)
plt.show()
```



```
[65]: ax=df.plot(x='SRA', y='pct_matched', kind='bar',figsize=(12,4))
ax.set_title(s_name, fontsize=12)
plt.show()
```



#### 0.0.11 Top nt database contigs matches per SRA

```
[66]: def get_sra_dict(sra):
total_dict={}
asc_desc={}
accessions, descriptions, counts, total=get_asc_descr_count(sra)
```

```

for asc,desc,cnt in zip(accessions,descriptions, counts):
    if asc in total_dict:
        total_dict[asc]+=int(cnt)
    else:
        total_dict[asc]=int(cnt)
    if asc not in asc_desc:
        asc_desc[asc]=desc
return total_dict, asc_desc, total

```

```

[67]: for sra in sra_list:
    total_dict, asc_desc, total= get_sra_dict(sra)
    print(f'{sra}, number of contigs {total}')
    print_top_n_sp(total_dict, max_num=10)
    print('\n')

```

```

SRR11093265, number of contigs 10
Pangolin coronavirus isolate MP789, complete genome : 14
Pangolin coronavirus isolate MP789 genomic sequence : 10
Pangolin coronavirus isolate cDNA8-S surface glycoprotein (S) gene, complete cds
: 3
Pangolin coronavirus isolate cDNA9-S surface glycoprotein (S) gene, complete cds
: 3
Pangolin coronavirus isolate cDNA16-S surface glycoprotein (S) gene, complete
cds : 3
Pangolin coronavirus isolate cDNA18-S surface glycoprotein (S) gene, complete
cds : 3
Pangolin coronavirus isolate cDNA20-S surface glycoprotein (S) gene, complete
cds : 3
Pangolin coronavirus isolate cDNA31-S surface glycoprotein (S) gene, complete
cds : 3
Severe acute respiratory syndrome coronavirus 2 isolate 230077 genome assembly,
chromosome: 1 : 1
Synthetic construct clone C35/41, complete sequence : 1

```

```

SRR11093266, number of contigs 34
Pangolin coronavirus isolate PCoV_GX-P1E, complete genome : 117
Pangolin coronavirus isolate PCoV_GX-P2V, complete genome : 95
Pangolin coronavirus isolate PCoV_GX-P4L, complete genome : 90
Pangolin coronavirus isolate PCoV_GX-P5L, complete genome : 74
Pangolin coronavirus isolate PCoV_GX-P5E, complete genome : 72
Pangolin coronavirus isolate PCoV_GX-P3B genomic sequence : 72
Lutra lutra genome assembly, chromosome: 16 : 2
Manis javanica isolate MP_PG03-UM mitochondrion, complete genome : 1
Mus musculus clone contig 1 chromocenter region genomic sequence : 1
Mus musculus clone contig 5 chromocenter region genomic sequence : 1

```



SRR11093267, number of contigs 100  
Pangolin coronavirus isolate PCoV\_GX-P5L, complete genome : 449  
Pangolin coronavirus isolate PCoV\_GX-P1E, complete genome : 366  
Pangolin coronavirus isolate PCoV\_GX-P2V, complete genome : 347  
Pangolin coronavirus isolate PCoV\_GX-P5E, complete genome : 338  
Pangolin coronavirus isolate PCoV\_GX-P4L, complete genome : 326  
Pangolin coronavirus isolate PCoV\_GX-P3B genomic sequence : 126  
PREDICTED: Manis pentadactyla uncharacterized LOC118928956 (LOC118928956), ncRNA : 3  
Manis javanica isolate MP\_PG03-UM mitochondrion, complete genome : 2  
Manis javanica isolate EP2 mitochondrion, complete genome : 2  
Canis lupus familiaris breed Labrador retriever chromosome 30b : 2

SRR11093268, number of contigs 10  
Pangolin coronavirus isolate PCoV\_GX-P5E, complete genome : 43  
Pangolin coronavirus isolate PCoV\_GX-P5L, complete genome : 40  
Pangolin coronavirus isolate PCoV\_GX-P2V, complete genome : 37  
Pangolin coronavirus isolate PCoV\_GX-P1E, complete genome : 31  
Pangolin coronavirus isolate PCoV\_GX-P4L, complete genome : 23  
Pangolin coronavirus isolate PCoV\_GX-P3B genomic sequence : 11  
Lutra lutra genome assembly, chromosome: 16 : 1  
PREDICTED: Manis javanica phosphatidylinositol glycan anchor biosynthesis class Q (PIGQ), transcript variant X7, mRNA : 1  
PREDICTED: Manis javanica uncharacterized LOC118969509 (LOC118969509), ncRNA : 1  
PREDICTED: Canis lupus familiaris collagen alpha-1(I) chain-like (LOC119879328), misc\_RNA : 1

SRR11093269, number of contigs 223  
Pangolin coronavirus isolate PCoV\_GX-P4L, complete genome : 261  
Pangolin coronavirus isolate PCoV\_GX-P5L, complete genome : 259  
Pangolin coronavirus isolate PCoV\_GX-P1E, complete genome : 257  
Pangolin coronavirus isolate PCoV\_GX-P2V, complete genome : 255  
Pangolin coronavirus isolate PCoV\_GX-P3B genomic sequence : 253  
Pangolin coronavirus isolate PCoV\_GX-P5E, complete genome : 173  
Megaerops niphanae clone Mnip32b LINE-1 retroposon ORF II pseudogene, partial sequence : 5  
Manis tricuspis clone Mtri123b LINE-1 retroposon ORF II pseudogene, partial sequence : 4  
Manis tricuspis clone Mtri131b LINE-1 retroposon ORF II pseudogene, partial sequence : 3  
Canis lupus familiaris breed Labrador retriever chromosome 30b : 2

SRR11093270, number of contigs 11832

Pangolin coronavirus isolate PCoV\_GX-P3B genomic sequence : 16  
 PREDICTED: Manis javanica proline rich coiled-coil 2A (PRRC2A), mRNA : 15  
 Pangolin coronavirus isolate PCoV\_GX-P1E, complete genome : 14  
 PREDICTED: Manis javanica LDL receptor related protein 1 (LRP1), transcript variant X3, mRNA : 14  
 PREDICTED: Manis javanica LDL receptor related protein 1 (LRP1), transcript variant X4, mRNA : 14  
 Pangolin coronavirus isolate PCoV\_GX-P4L, complete genome : 13  
 PREDICTED: Manis javanica LDL receptor related protein 1 (LRP1), transcript variant X1, mRNA : 13  
 PREDICTED: Manis javanica LDL receptor related protein 1 (LRP1), transcript variant X2, mRNA : 13  
 Erysipelothrix rhusiopathiae strain KC-Sb-R1 chromosome, complete genome : 12  
 Pangolin coronavirus isolate PCoV\_GX-P5L, complete genome : 12

SRR11093271, number of contigs 74  
 Pangolin coronavirus isolate PCoV\_GX-P2V, complete genome : 11  
 Mycoplasma arginini strain HAZ145\_1 : 10  
 Mycoplasma arginini strain NCTC10129 chromosome 1 : 8  
 Mycoplasma hyorhinis strain NCTC10130 genome assembly, chromosome: 1 : 4  
 Mycoplasma hyorhinis strain NCTC10121 genome assembly, plasmid: 3 : 4  
 Mycoplasma hyorhinis strain JF5820 chromosome, complete genome : 4  
 Mycoplasma hyorhinis strain IMT49388 chromosome, complete genome : 4  
 Mycoplasma hyorhinis HUB-1, complete genome : 4  
 Mycoplasma hyorhinis GDL-1, complete genome : 4  
 Mycoplasma hyorhinis SK76, complete genome : 4

[68]: *### Get specific contigs matching a species/name*

```

[69]: def write_contigs(sra, match_names):
    accessions, descriptions, counts, total=get_asc_descr_count(sra)
    asc_matches=[]
    for m in match_names:
        for a,d in zip(accessions, descriptions):
            if m.lower() in d.lower():
                asc_matches.append(a)
    gi_matches=[]
    for a in asc_matches:
        idx=ACCESSIONS.index(a)
        gi=GIS[idx]
        gi_matches.append(gi)
    contigs=[]
    path = BASE_PATH+sra+'/magic_blast/'
  
```

```

subset_f = open(path+f'{sra}_{match_names[0].replace(" ", "_", 1)}_subset_{f_contigs_file_tail}', 'w')
with open(path+f'{sra}_{f_contigs_file_tail}', 'r') as f:
    lines = [line for line in f]
    for line in lines:
        for gi in gi_matches:
            if gi in line:
                parts=line.split('\t')
                idx=GIS.index(gi)
                asc=ACCESSIONS[idx]
                t=TITLES[idx]
                ps=parts[:2]
                pe=parts[3:]
                ps.append(asc+' '+t.rstrip('\n'))
                parts=ps+pe
                p='\t'.join(parts)
                subset_f.write(p)
subset_f.close()

```

```
[ ]:
```

```
[70]: #set_accessions() #time consuming, comment out if already ran
      #assert ACCESSIONS is not None

```

```
[71]: #match_names=['vector']
      #for sra in sra_list:
      #    write_contigs(sra, match_names)

```

```
[72]: #match_names=['plasmid']
      #for sra in sra_list:
      #    write_contigs(sra, match_names)

```

```
[73]: #match_names=['synthetic']
      #for sra in sra_list:
      #    write_contigs(sra, match_names)

```

```
[74]: #match_names=['virus']
      #for sra in sra_list:
      #    write_contigs(sra, match_names)

```

```
[ ]:
```